UNIVERSITÀ
DEGLI STUDI
FIRENZE

The Byzantine general problem

DISTRIBUTED REAL TIME CYBER- PHYSICAL SYSTEMS

Rebeca Sal Carro – 7024022

Hugo Pérez Rodríguez – 7029395

Academic Year 2019/2020

Index

Requirements

Environment:

E01. The environment shall be reflective.

E02. The environment shall have dimensions of 60x50 cm.

E03. The environment shall be smooth.

Game Rules:

G01. At the beginning, a random number is generated to determine the ID of the general.

G02. The kilobot with the general's id will have the ORANGE color. The rest of the kilobots will have the RED color.

G03. Another random number is generated to know whether there will be a traitor or not.

G04. If the number generated for the traitor is between 4 and 7 (both included) there will be no traitor.

G05. If the number generated for the traitor is between 0 and 3 (both included), the kilobot with the id equal to the traitor's generated number will be the traitor.

G06. The traitor will have the BLUE color.

G07. If the general is also the traitor, the kilobot will be BLUE instead of ORANGE.

G08. The general is the first kilobot that moves.

G09. When the moving kilobot wants to transmit a message, it will change its color, depending on its message.

G10. If a moving kilobot wants to send an "attack" message it will change its color to GREEN for few seconds.

G11. If a moving kilobot wants to send a "no attack" message it will change its color to YELLOW for a few seconds.

G12. The moving kilobot will be moving until it sends a message to each of the others kilobots.

G13. When the moving kilobot transmits a message to each of the stationary kilobots, it changes his state and remains stopped until the end of the game.

G14. When a stationary kilobot receives a message, it saves the received message in a personal matrix in the position of the sender's id.

G15. The id of the next kilobot that moves is calculated after the current moving kilobot send its 3 messages.

G16. When all the kilobots have send their message, each kilobot counts the "attack" and "no attack" messages it has received from the others.

G17. If the kilobot had received more "attack" message than "no attack" message, its final decision will be "attack".

G18. If the kilobot had received more "no attack" message than "attack" message, its final decision will be "no attack".

G19. If the final decision of a kilobot is "attack" it will change its color to WHITE.

G20. If the final decision of a kilobot is "no attack" it will turn off its led and no color will be displayed.

G21. All kilobots except the traitor will take the same decision and act accordingly.

Kilobots

K1. At the beginning, the kilobots shall be in a square position.

K2. The kilobots are collocated according to their id's number.

K3. There shall be 4 kilobots.

K4. The general and the traitor (in case there is a traitor) are decided with a random number at the beginning of the program.

K5. When a kilobots wants to send an "attack" message, its color will be green. On the other hand, if the message is "no attack", its color will be yellow.

K5. The kilobots will save in a matrix all the received messages.

K6. When a kilobot has send one message to each of the others, it changes its state and remains stationary.

K7. When all the kilobots finish their moving phase, they will take a decision considering their received messages.

Design

The Byzantine Generals application was designed in the Blockly4SOS application with the following structure.

1. **Requirements**: as described previously.
2. **Environment:** a reflective and smooth whiteboard, affecting the kilobots communication.
3. **SoS:** describing the components of the system.
4. **Sequence diagrams:** describing the flow of the actions of the main program. Sequence diagrams will be explained in detail below.

The SoS is composed of three types of constituent system (CS): the OHC, the general and the lieutenants:

- The OHC interacts with 4 kilobots. The interactions happen through a RUPI (infrared) and a RUMI that can communicate some setup information to the bots, such as their ID. It is controlled by a human user, who starts the system. It is a closed system, as it doesn't have an autonomy.
- The General and the lieutenant have some common features, such as an infrared-based RUPI and a short-range vision around them, which limits their communication. The General and the lieutenant have their own RUMI to send messages to each other and some state variables. These two components are autonomous.

Query Diagram

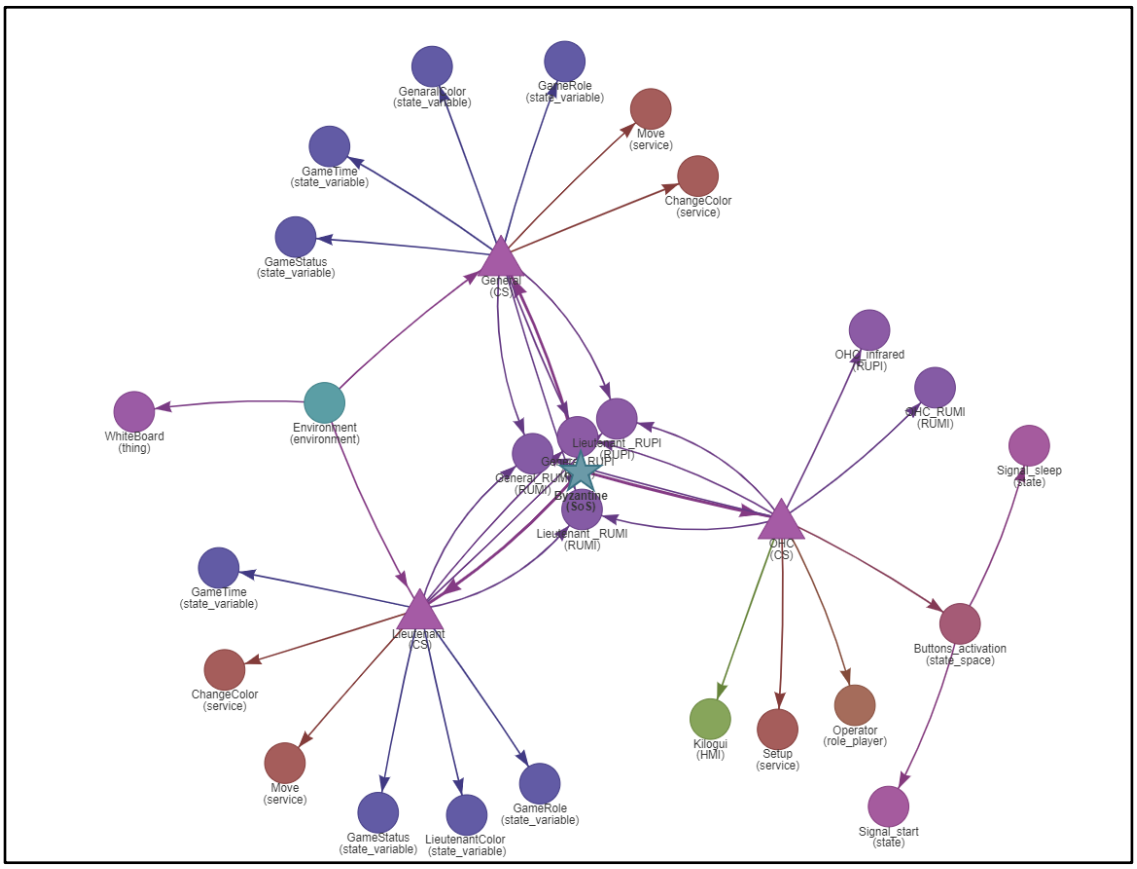

Figure 1. Query Diagram.

Sequence Diagram

To describe the sequence of the application, three different diagrams have been drawn.

- Setup Diagram.

  In this first phase OHC send all the main information to General and Lieutenants, information like their id, their role or their colour. This information is vital for the good functioning of the application. In addition, OHC is the component that start the application.

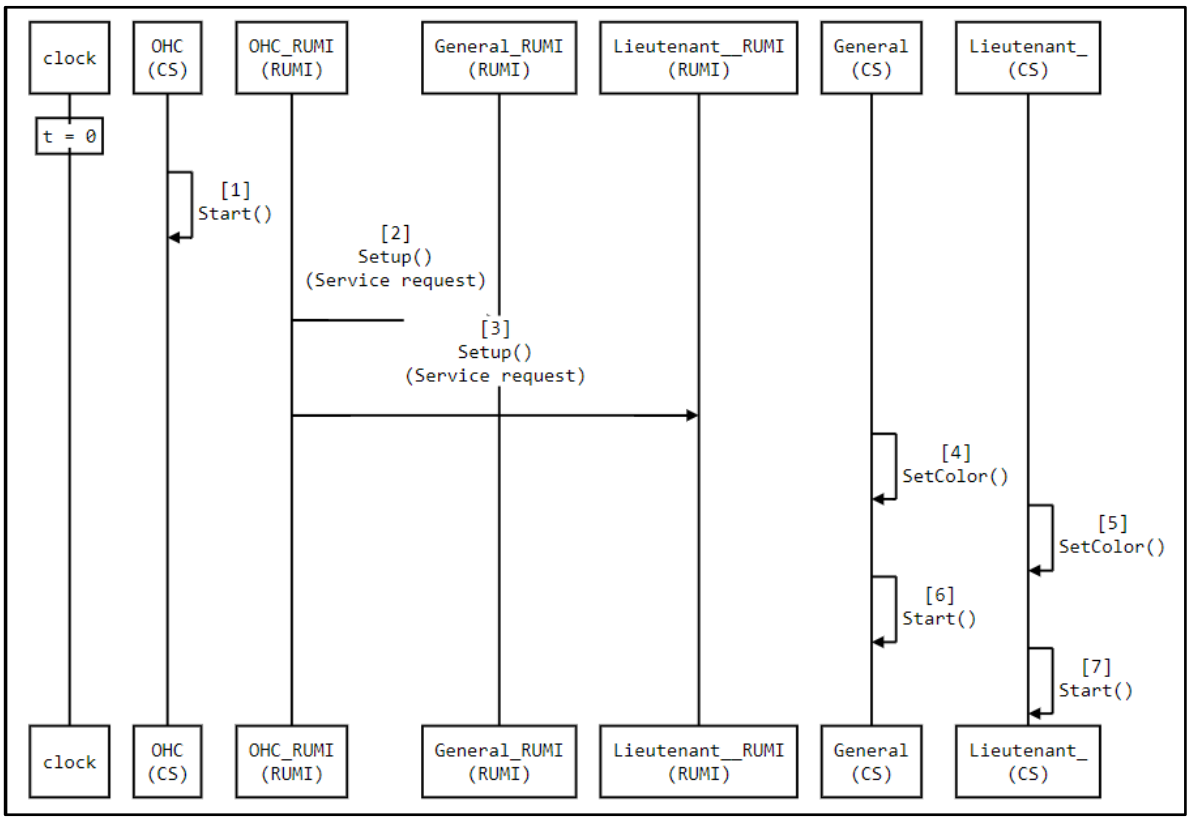

Figure 2. Setup Diagram.

- General and Lieutenant Diagrams.

  These two diagrams show the main functions of the General and lieutenants: send messages, move, change their colour and, firstly, send information across the RUMIs.

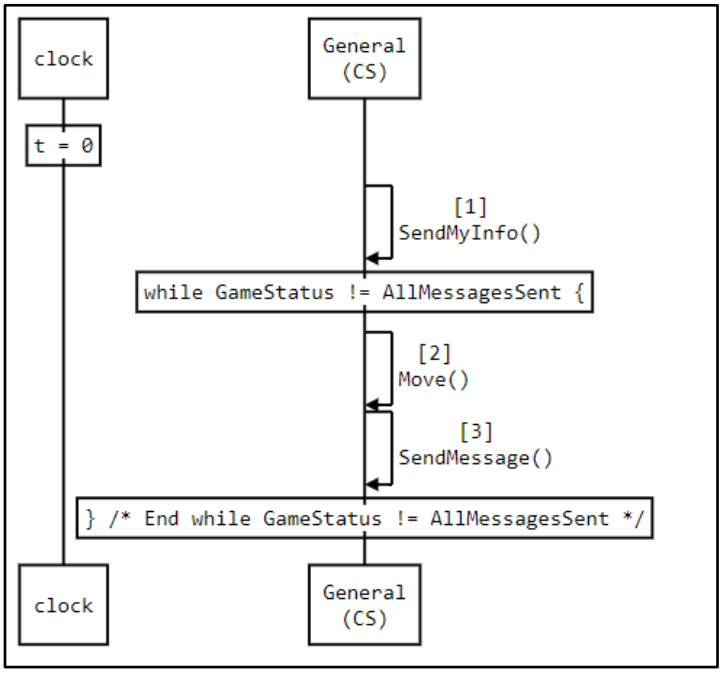

Figure 3. General Diagram.

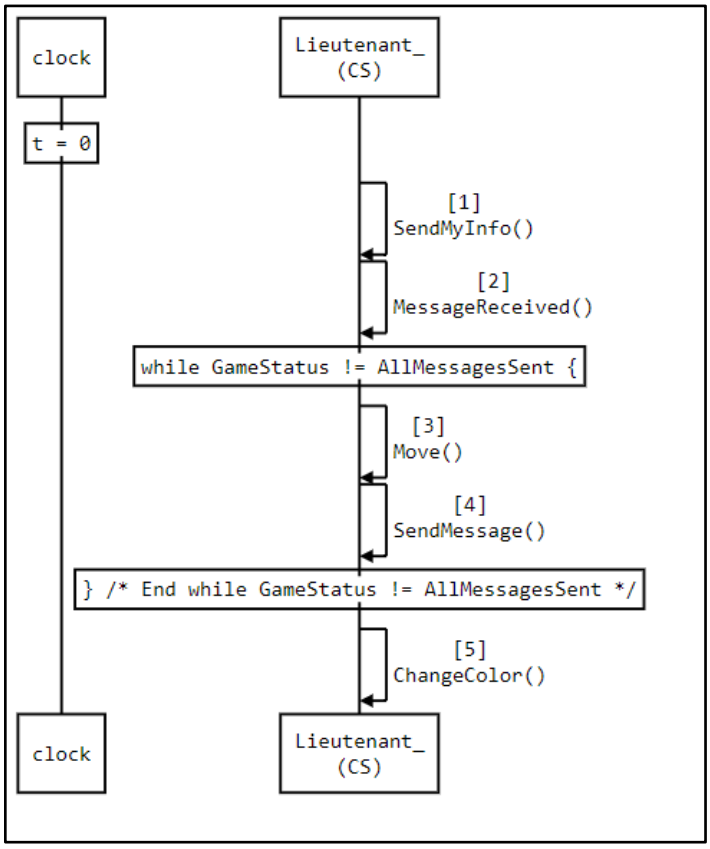

Figure 4. Lieutenant Diagram.

Implementation

The project can be found at the following URL: https://github.com/rsalca01/Byzantine.git

The project is structured in different modules, so as to handle different functions:

- The main module.
- The initialization module, which takes care of initializing all the data variables that the kilobots will use during the program.
- The time module, managing the local logical time of the kilobots.
- The movement module, in charge of the moving the kilobots to each of the corners of the square.
- The communication module, that contains all the functions related to sending and receiving messages.
- The colors module, responsible for the changing the color of the kilobots every time they send a message or when they reach a final decision.
- The game module, which contains the functions that analyze the current state of each kilobot and determine which will be the following movement for each kilobot.

Simulation

The simulation of this project was done on the Kilombo simulator.

If you want to build the program you just have to open a terminal in the working directory and run the following command:

>make -f Makefile.osx

```
MacBook-Pro-de-Rebeca:src rebeca$ make -f Makefile.osx
```

The program is executed by opening a terminal in the working directory, which contains the kilombo.json file and running the following command:

>./byzantine -b position.json

```
MacBook-Pro-de-Rebeca:Desktop rebeca$ cd DRTCPS-FinalProject/src
MacBook-Pro-de-Rebeca:src rebeca$ ./byzantine -b position.json
```

The initial setup for the simulation has the following settings:

- Number of bots: 4.
- Positions at the start: the kilobots are placed in each of the corners of a square.

After the execution we can have two different scenarios:

1: Without a traitor.

In this scenario, after the execution the general is orange and the lieutenants are red.

The first kilobot to move will be always the general. The general kilobot moves to the position of the other three lieutenants sequentially to send each of them a message: to attack or not to attack. Because there is no traitor, the other three kilobots will send each other the same message.

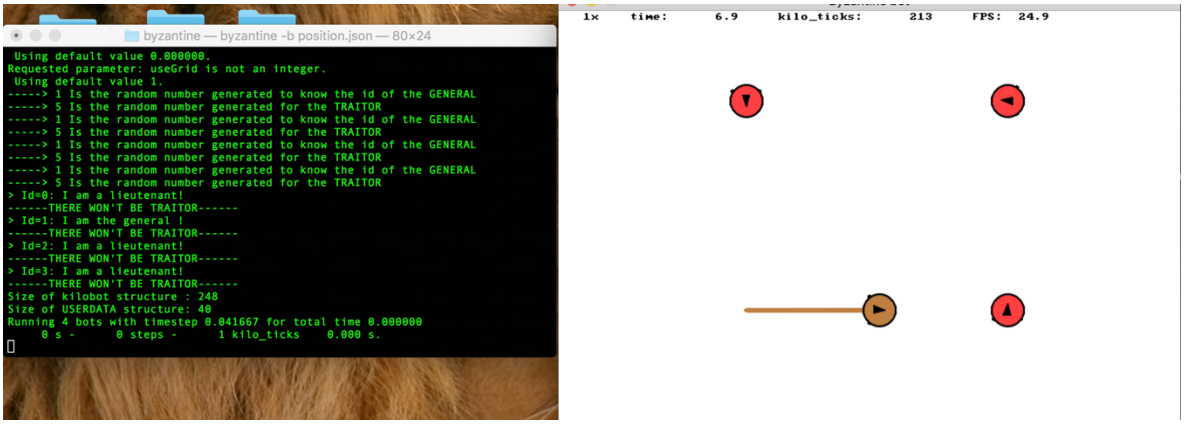

Figure 5. The general kilobot (orange) is the first one to start moving to each of the lieutenants (red).

When the moving kilobot detects a stopped kilobot, the moving kilobot stops and changes its color depending on the message it wants to send: green for attack and yellow for no attack.

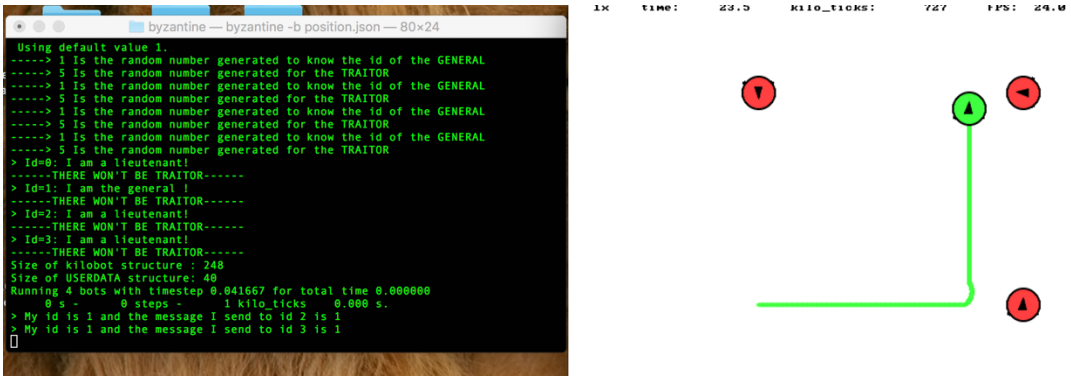

Figure 6. The general kilobot turns green in front of each lieutenant to send them an “attack” message.

The moving kilobot will continue doing the square until it has sent a message to each of the stopped kilobots.

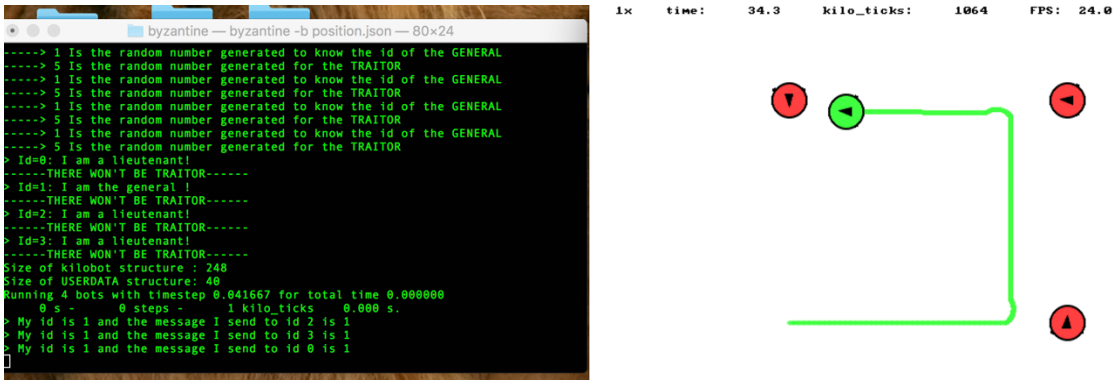

Figure 7. The general sends the message to the three lieutenants. Having finished its round, all three lieutenants will have saved this message in their matrix. This is the message they will send each other in the next three rounds.

When a stopped kilobot receives a message, it saves the message in a matrix in the position of the sender kilobot's id.

After a kilobot has completed its round, the identity number of the next kilobot to start moving will be automatically computed. Thus, it is decided which kilobot will start moving next.

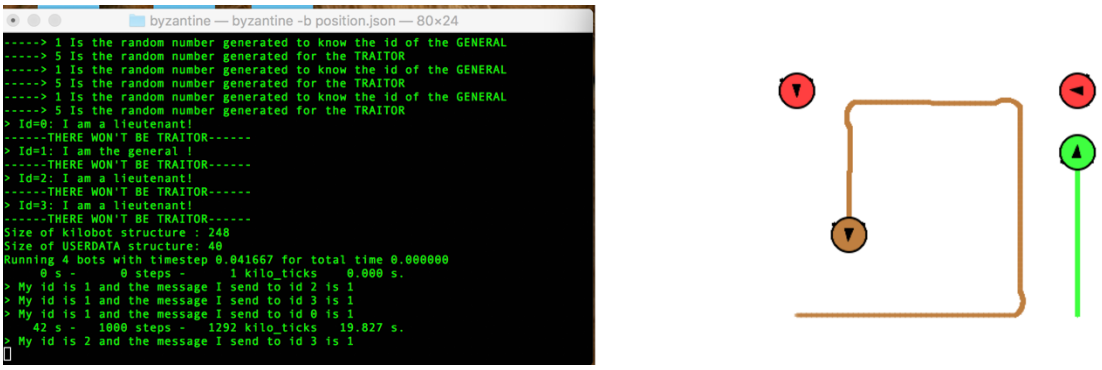

Figure 8. The next kilobot starts moving and transmits the "attack" message (green).

The message sent by the lieutenants to each other will always be the message sent by the general to each of them, as there is no traitor.

In the end, when all the kilobots have sent the message to the lieutenants, each lieutenant will have stored three messages in their matrix. In order for the kilobot to reach a decision (attack or not), the matrix is analyzed to calculate the number of "attack" and "not attack" messages in it. A majority of attack messages will result in the kilobot attacking, so it will change its

color to white. On the other hand, a majority of "not attack" messages will result in the kilobot not attacking, therefore turning off its led light.

All the kilobots end up making the same decision and act coordinately because they only have one type of message in the matrix (the one send by the general initially).

Figure 9. All the kilobots have reached a decision according to the initial message sent by the general. Thus, they are attacking and turn white.

2. With a traitor.

In this case, after the execution of the program, one of the kilobots will be a traitor (blue-colored). If the general is also the traitor, there will only be one blue kilobot and none of the kilobots will be orange.

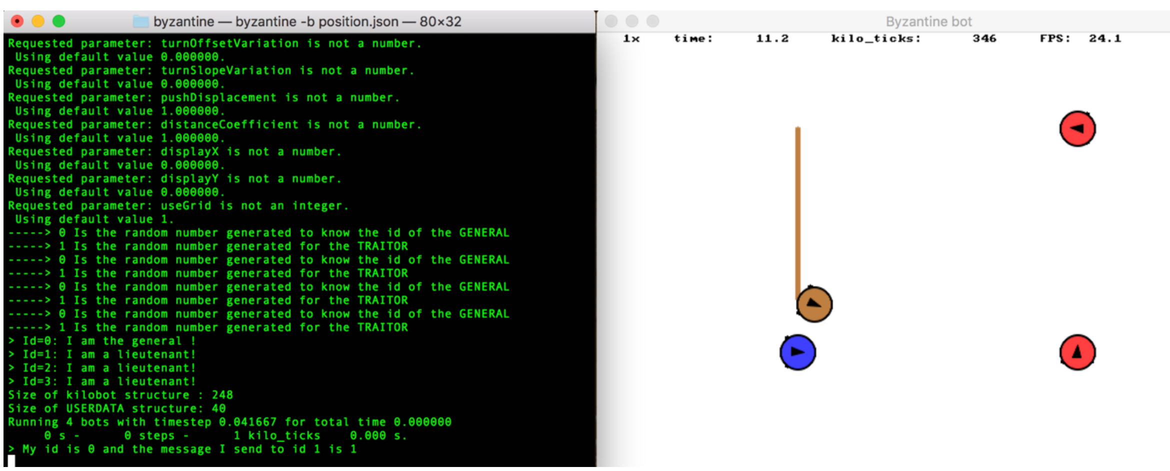

Figure 10. There are three types of kilobot: the general (brown), the traitor (blue) and the loyal lieutenants (red). In this case the traitor and the general are two different kilobots. The general starts its round first.

The movement of the kilobots while the program is running is the same as if there was no traitor.

In this case the only difference is that when the traitor wants to send a message to the other kilobots, it will send the opposite of the message received from the general.

If the general is the traitor, it will send different messages to each lieutenant.

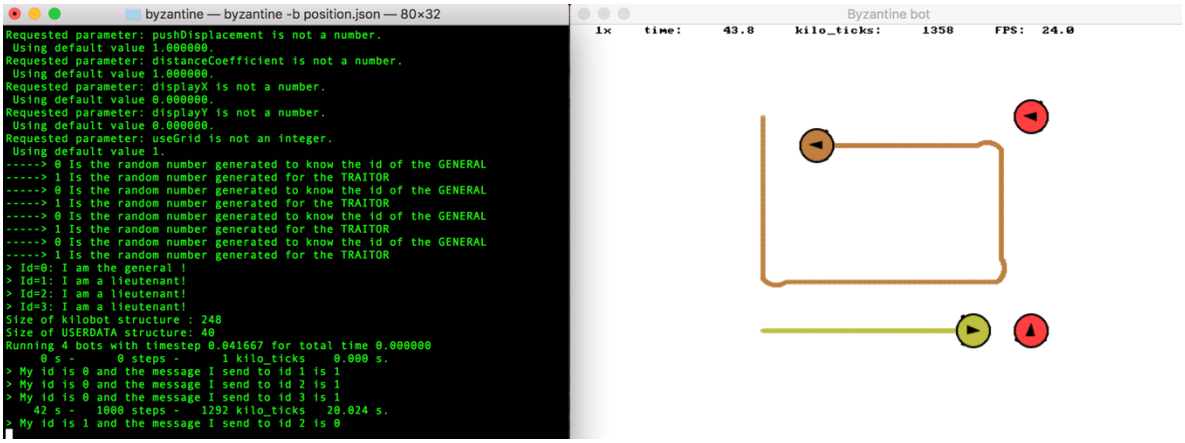

Figure 11. After the general finishes, the next kilobot starts moving. In this case, the next kilobot is a traitor. Because the traitor received an "attack" message from the general, it will send a "not attack" message to the other kilobots. As can be seen in the terminal, the general (id 0) sent a message 1 (attack), which makes the traitor (id 1) send a message 0 to the other kilobots.

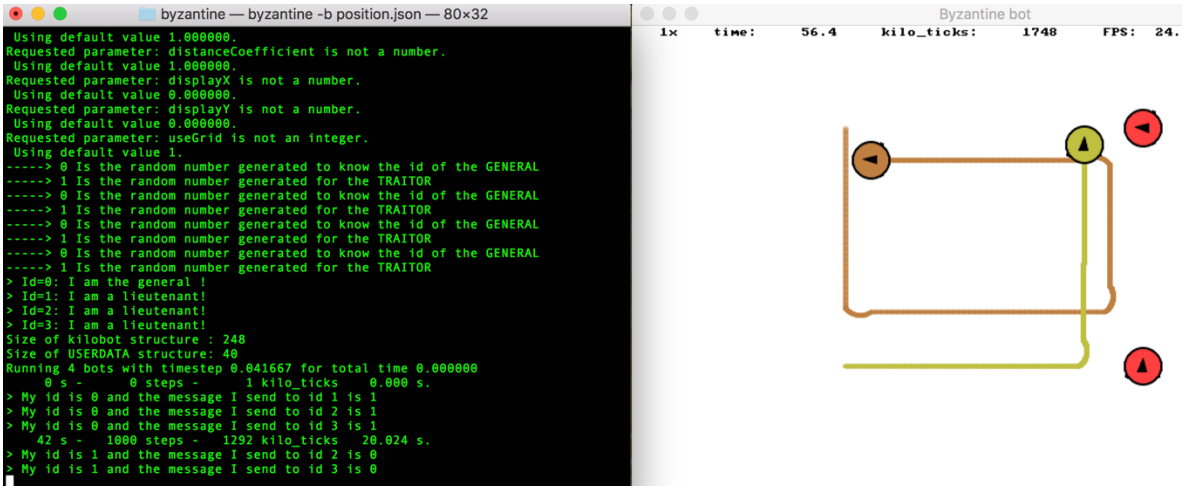

Figure 12. The traitor sends a "not attack" message to kilobot 3 (id 3).

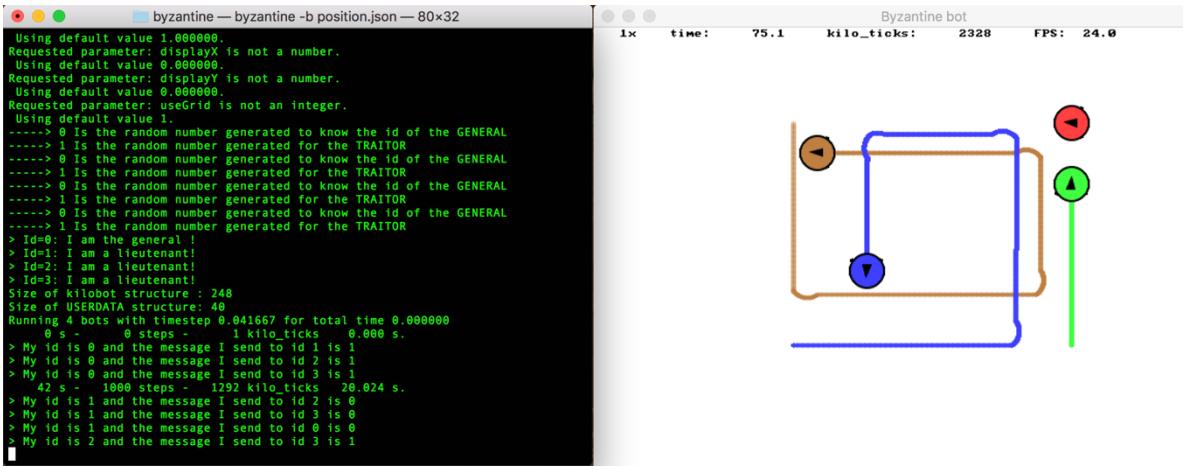

Figure 13. The third kilobot (id 2), which is loyal, sends an "attack" message (green) to the other two lieutenants. This loyal lieutenant is sending the first message it received, which is always the one received by the general.

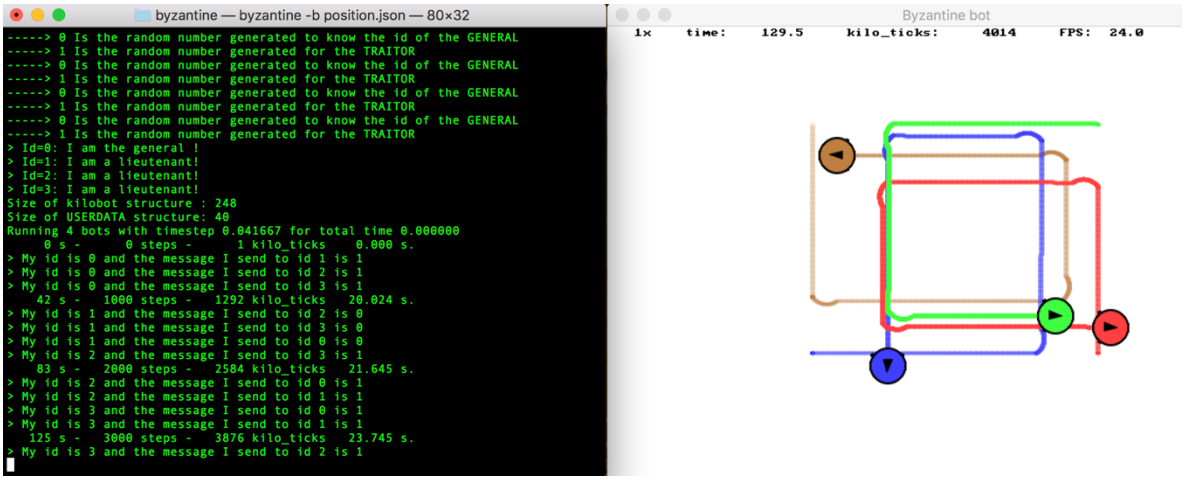

Figure 14. The last kilobot (id 3), also loyal, sends an "attack" message (green) to the other two lieutenants. This loyal lieutenant is sending the first message it received, which is always the one received by the general.

After having analyzed the matrices all the kilobots will reach a decision, except for the traitor which will have its initial color.

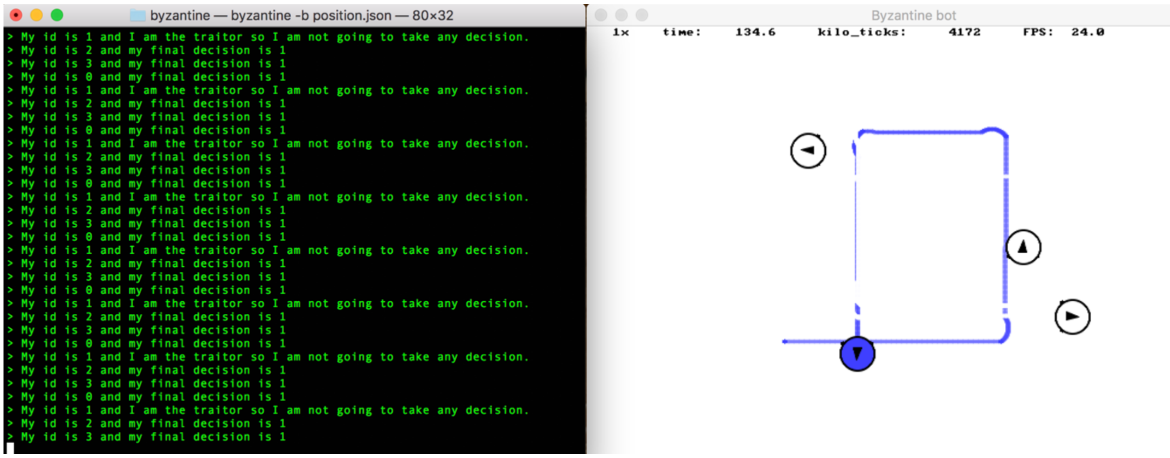

Figure 15. All kilobots but the traitor made the decision of attacking, in accordance with the messages sent by the loyal kilobots. The matrices of all the kilobots have stored two "attack" messages and a "not attack" message.

In the end, all the kilobots (except for the traitor) will end up making the same decision because, as there could only be one traitor, at least two thirds of the messages in the matrix will be the same for all kilobots.